Curso: Java COMPLETO - Programação Orientada a Objetos + Projetos
http://educandoweb.com.br
Prof. Dr. Nelio Alves

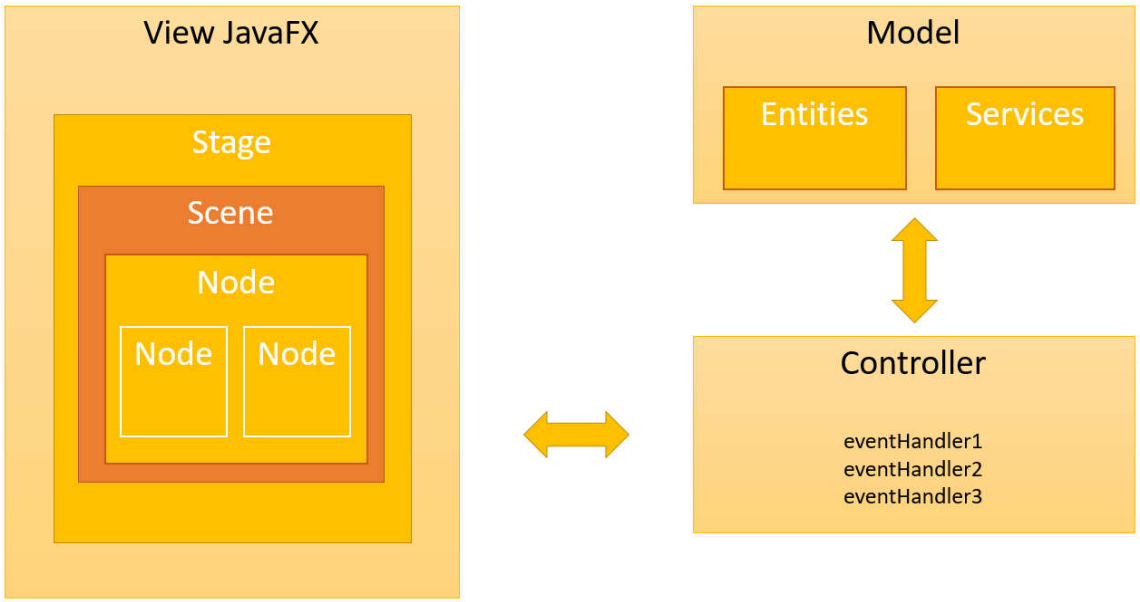
Capítulo: Interface Gráfica com JavaFX

Objetivo geral:

- Conhecer os fundamentos e a estrutura do JavaFX
- Conhecer os principais componentes visuais do JavaFX

Visão geral do JavaFX e MVC

- JavaFX é o sucessor do **Swing** e **Java AWT** para interfaces gráficas com Java
- JavaFX pode ser usado para desktop, web e mobile
- Uma tela JavaFX pode ser montada via código Java, ou via código FXML
- Com o lançamento do Java 11, JavaFX não é mais parte do JDK
  - O JavaFX precisa ser baixado e configurado separadamente
  - É mantido pela Gluon: https://gluonhq.com/products/javafx/
- JavaFX é projetado sobre o padrão MVC
  - Model - consiste nos dados de domínio e toda lógica de transformação desses dados
  - Views - São as telas de interação com o usuário (UI)
  - Controllers - São as classes responsáveis por tratar as interações do usuário com as views (manipulação de eventos de interação com as telas)



- Hierarquia do JavaFX: https://docs.oracle.com/javase/8/javafx/api/overview-tree.html

Instalação do Scene Builder

Nota: a Oracle disponibiliza o código fonte do Scene Builder. Os builds para instalação são mantidos pela Gluon.

- Scene Builder: https://gluonhq.com/products/scene-builder/

Preparação do Eclipse

ATENÇÃO: Eclipse 4.9 ou superior: http://www.eclipse.org/downloads/packages/

Checklist:

- Baixar o JavaFX SDK: https://gluonhq.com/products/javafx/
    - Salvar em uma pasta "fácil", de preferência com nome sem espaços
        - Exemplo: `C:\java-libs`

- Instalar o plug-in **E(fx)clipse** no Eclipse (**ATENÇÃO**: versão 3.4.1 ou superior)
    - Help -> Install new Software
    - Work with: escolha o correspondente à versão do seu Eclipse
        - Exemplo: 2018-09 - http://download.eclipse.org/releases/2018-09
        - Exemplo de link direto: http://download.eclipse.org/efxclipse/updates-released/3.4.1/site/
    - Localize e(fx)clipse
    - Next Next etc.
    - Reinicie o Eclipse

- Referenciar o SceneBuilder no Eclipse:
    - Window -> Preferences -> JavaFX
    - Entrar o caminho completo do arquivo executável do Scene Builder
        - Exemplo: C:\Users\Nelio\AppData\Local\SceneBuilder\SceneBuilder.exe

- Criar uma User Library no Eclipse com o nome de JavaFX:
    - Window -> Preferences -> Java -> Build Path -> User Libraries -> New
    - Dê o nome de User Library
    - Add External Jars (aponte para a pasta bin do JavaFX)

Criando um novo projeto JavaFX no Eclipse

Checklist:

- Criação do projeto:
    - File -> New -> Other -> JavaFX Project
    - Dê um nome ao projeto e clique **Next**
    - Na aba Libraries, selecione Modulepath, clique Add Library, e selecione JavaFX
    - Clique **Finish**
    - Module Info: Don't Create

- Configuração do build:
    - Botão direito no projeto -> Run As -> Run Configurations -> Arguments -> VM Arguments
    - Copiar o conteúdo abaixo, adaptando para sua pasta:

```
--module-path C:\java-libs\javafx-sdk\lib --add-modules=javafx.fxml,javafx.controls
```

Código da classe Main.java:

```
package application;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;

public class Main extends Application {
	@Override
	public void start(Stage primaryStage) {
		try {
			BorderPane root = new BorderPane();
			Scene scene = new Scene(root,400,400);
			scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
			primaryStage.setScene(scene);
			primaryStage.show();
		} catch(Exception e) {
			e.printStackTrace();
		}
	}

	public static void main(String[] args) {
		launch(args);
	}
}
```

Testando o FXML

Projeto: https://github.com/acenelio/javafx1

Checklist:

- Criar um pacote **gui**
- Criar um FXML no projeto: Botão direito no pacote gui -> New -> Other -> New FXML Document
  - Nome: **View**
- Abra o FXML no SceneBuilder: Botão direito -> Open in SceneBuilder
- Observe as guias: Library, Document e Inspector
  - Inspector -> Layout: defina largura e altura, depois salve
  - Library -> Control: acrescente alguns controles (ex: TextField, Button)
- De volta ao Eclipse, na classe Main, refazer o método start:

```
@Override
public void start(Stage stage) {
	try {
		Parent parent = FXMLLoader.load(getClass().getResource("/gui/View.fxml"));
		Scene scene = new Scene(parent);
		stage.setScene(scene);
		stage.show();
	}
	catch (IOException e) {
		e.printStackTrace();
	}
}
```

Tratando eventos com JavaFX

Projeto: https://github.com/acenelio/javafx2

Checklist:

- Crie uma classe controladora da sua view (ex: ViewController.java)
- No controlador:
  - Criar um atributo correspondente ao controle desejado e anotá-lo com @FXML
  - Criar um método para tratar o evento desejado do controle e anotá-lo com @FXML

- Na view (Scene Builder):
  - Associar a view ao controller (aba Controller)
  - Selecione o controle e associe a ele o id (aba Code)
  - Selecione o controle e associe o método ao evento desejado (aba Code)

DICA: quando mudar algo no SceneBuilder, use **Project -> Clean** no Eclipse para forçar a atualização do projeto

Mostrando Alert

Projeto: https://github.com/acenelio/javafx3

```
package gui.util;

import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;

public class Alerts {

	public static void showAlert(String title, String header, String content, AlertType type) {
		Alert alert = new Alert(type);
		alert.setTitle(title);
		alert.setHeaderText(header);
		alert.setContentText(content);
		alert.show();
	}
}
```

Usando TextField e Label (app para calcular soma)

Projeto: https://github.com/acenelio/javafx4

Checklist:

- Desenhar tela no SceneBuilder (usar um label para resultado)
  - Propriedade "promptText"
- Criar um controller e implementar código para mostrar a soma
  - Tratar exceção NumberFormatException
- De volta ao Scene Builder, fazer as associações de id e evento

DICA: quando mudar algo no SceneBuilder, use **Project -> Clean** no Eclipse para forçar a atualização do projeto

Limitações para TextField e Initializable

Projeto: https://github.com/acenelio/javafx5

Referências: https://docs.oracle.com/javase/8/javafx/api/javafx/fxml/Initializable.html

Checklist:

- Criar classe utilitária Constraints
- Fazer o controlador implementar a interface **Initializable**

```
package gui.util;

import javafx.scene.control.TextField;

public class Constraints {

	public static void setTextFieldInteger(TextField txt) {
		txt.textProperty().addListener((obs, oldValue, newValue) -> {
		 if (newValue != null && !newValue.matches("\\d*")) {
			txt.setText(oldValue);
		 }
	    });
	}

	public static void setTextFieldMaxLength(TextField txt, int max) {
		txt.textProperty().addListener((obs, oldValue, newValue) -> {
		 if (newValue != null && newValue.length() > max) {
			txt.setText(oldValue);
		 }
	    });
	}

	public static void setTextFieldDouble(TextField txt) {
		txt.textProperty().addListener((obs, oldValue, newValue) -> {
			if (newValue != null && !newValue.matches("\\d*([\\.]\\d*)?")) {
		     txt.setText(oldValue);
		  }
		   });
	}
}
```

ComboBox

Projeto: https://github.com/acenelio/javafx6

Checklist:

- Propriedade Prompt Text
- Usar tipo genérico, por exemplo: ComboBox<Person>
- ObservableList<Person>, ObservableSet<Person>, ObservableMap<Person>
  - Para criar um ObservableList: FXCollections.observableList(list)
- ComboBox.setItems(observableList)
- Nota: o combo box, por padrão, mostra o toString do objeto
- Para obter o elemento selecionado: comboBox.getSelectionModel().getSelectedItem()

- Para acessar a coleção: comboBox.getItems()
- Para definir o que mostrar na comboBox:

```
Callback<ListView<Person>, ListCell<Person>> factory = lv -> new ListCell<Person>() {
    @Override
    protected void updateItem(Person item, boolean empty) {
        super.updateItem(item, empty);
        setText(empty ? "" : item.getName());
    }
};

comboBox.setCellFactory(factory);
comboBox.setButtonCell(factory.call(null));
```

Visão geral dos principais containers de layout

Checklist:

- AnchorPane
- GridPane
- SplitPane
- VBox & HBox
- BorderPane
- ScrollPane